AD-A066 851 NAVAL UNDERWATER SYSTEMS CENTER NEW LONDON CONN NEW --ETC F/G 17/1
A PROGRAM TO SIMULATE BUTTERWORTH FILTERS ON A DIGITAL COMPUTER--ETC(U)
OCT 70 S R VAN DER VEEN

UNCLASSIFIED NUSC/NL-TM-2211-309-70 NL

1 OF
AD A
066851

END
DATE
FILMED
6-79
DDC

NUSC/NL Problem No.

A-407-00-00
SF 11 552 002-12862

(9) Technical memo.,

NEW LONDON LABORATORY
NAVAL UNDERWATER SYSTEMS CENTER
NEW LONDON, CONNECTICUT 06320

(6) A PROGRAM TO SIMULATE BUTTERWORTH FILTERS ON A DIGITAL COMPUTER.

by

(10) Steven R. van der Veen

(14) NUSC/NL Technical Memorandum No. 2211-309-70
TM-2211-309-70

(11) 1 October 1970

(12) 25p.

(16) F11552

(17) SF1155200 2

INTRODUCTION

Filtering of underwater acoustic data is an operation which is typically performed in an analog manner. This operation is limited in flexibility due to the quantity and availability of hardware and limited in accuracy because of component tolerances, aging and environmental deterioration. It is the purpose of this memorandum to discuss the processing of broad band acoustic data using digital computer filtering on digitized data. This procedure is limited by the need for sampling rates several times the maximum frequency present in the signal, but bypasses the above limitations by allowing the simulation of a large number of filters. The accuracy is improved and further computer processing can be applied to the results.

ADMINISTRATIVE INFORMATION

This memorandum was prepared under NUSC/NL Project Title: Forward Scattering Studies For Sonar Design and Performance Prediction, S. R. Santaniello, Principal Investigator. The sponsoring activity was Naval Ships Systems Command, Code OOVI, J. Reeves, Program Manager.

While the computer programs are presently implemented to simulate only Butterworth type low pass (LP) and band pass (BP) filter, they are general enough to allow expansion to include any filter which is representable in terms of a finite pole-zero transfer function configuration. The formulation presented here produces, under mild constraints, digital filters with the same impulse response as the equivalent analog filter.

THEORY

Using z-transform notation (ref a and b), a time

$$x(kT) \quad , k=0,1,2...$$

can be expressed as a function of a complex variable z by the transform:

$$(1) \quad X(z)= \sum_{k=0}^{\infty} x(kT)\cdot z^{-k}$$

A linear filtering operation, such as that represented by the finite difference equation:

$$\sum_{i=0}^{N} a_i \cdot y(k-i) = \sum_{j=0}^{M} b_j \cdot x(k-j) \quad , k=0,1,2...$$

can be represented in z-transform notation by the equation:

$$(1a) \quad Y(z) = H(z)\cdot X(z) \ , \ \text{where} \quad H(z)= \frac{\sum_{j=0}^{M} b_j z^{-j}}{\sum_{i=0}^{N} a_i z^{-i}}$$

This equation is analogous to the continuous-time filtering operation represented in Laplace transform notation as:

$$(2) \quad Z(s)=H(s)\cdot X(s)$$

Letting the z-transform of a sampled time function x(t) be denoted $X^*(z)$ and letting the sampling operator on a Laplace transform be denoted by $(x(s))^*$ (so that $(X(s))^*=X^*(z)$), we can examine the conditions that are sufficient to cause the sampled output of a filter to be equal to the output of the equivalent digital filter.

We will assume that $X^*(z)$ and $H^*(z)$ are equal to X(z) and H(z). That is, the z-transforms used in evaluating the output of the digital filter were derived by sampling the input to and impulse response of the analog filter, or by an equivalent method.

The relationship between Z(s) and Z*(z) is derived in two forms by Freeman(a). They are:

(3) $Z^*(z) = \sum \text{residues of } \frac{Z(s)}{1-e^{sT}z^{-1}} \text{ at the poles of } Z(s).$

and where T^{-1} is the sampling frequency

(4) $Z^*(z) = \frac{1}{T} \sum_{l=-\infty}^{\infty} Z\left(s + \frac{2\pi i l}{T}\right)$, $i = \sqrt{-1}$, $z = e^{sT}$

Applying the sampling operator to eq.(2), we have:

$$Z^*(z) = (H(s) \cdot X(s))^*$$

Substituting from eq.(4), we get:

(4a) $$Z^*(z) = \frac{1}{T} \sum_{l=-\infty}^{\infty} \left[H\left(s + \frac{2\pi i l}{T}\right) \cdot X\left(s + \frac{2\pi i l}{T}\right) \right]$$

If both X and H are bandlimited (i.e. if X(s) and H(s) are zero for all s such that $|s| > \pi$), then: (4a) reduces to:

$$Z^*(z) = T \left(\frac{1}{T} \sum_{l=-\infty}^{\infty} H\left(s + \frac{2\pi i l}{T}\right) \right) \cdot \left(\frac{1}{T} \sum_{l=-\infty}^{\infty} X\left(s + \frac{2\pi i l}{T}\right) \right)$$

$$= T \cdot H^*(z) \cdot X^*(z) = T \cdot H(z) \cdot X(z)$$

Thus, if both X and H are bandlimited to less than half the sampling frequency, then the sampled output of the analog filter will be equal to the output of the digital filter amplified by the sampling period.

NUMERICAL METHOD

Many types of filters can be represented in terms of the pole-zero configuration of their transfer functions in the complex frequency plane.

Their transfer functions can then be written:

(5) $$H(s) = K \left[\frac{\prod_{i=1} (s - r_i)}{\prod_{j=1}^{N} (s - p_j)} \right]$$

Using eq.(3) reduces (5) to:

$$H^*(z)= K \sum_{j=1}^{N} \frac{\prod_{i=1}^{M}(P_j - C_i)}{(1-e^{P_j T} z^{-1}) \prod_{\substack{\ell=1 \\ \ell \neq j}}^{N}(P_j - P_\ell)}$$

in the z-domain. Letting:

$$A_j = e^{p_j T} \quad \text{and} \quad K_j = K \frac{\prod_{i=1}^{M}(P_j - C_i)}{\prod_{\ell=1, \ell \neq j}^{N}(P_j - P_\ell)}$$

we have a sum of 1st order terms:

$$H(z)= \sum_{j=1}^{N} \frac{K_j}{(1-A_j z^{-1})}$$

The K_j and A_j will either be real or in complex conjugate pairs. Each conjugate pair can be combined to yield second order terms of the type:

$$\frac{C_{j1} - C_{j2} \cdot z^{-1}}{1-B_{j1} \cdot z^{-1} + B_{j2} \cdot z^{-2}}$$

Thus, H(z) can be represented as the sum of first and second order terms of the type seen in (1a), representing a difference equation in z-transform notation.

The second order terms are thus realized by the equations:

$$y_j(k) - B_{j1} \cdot y_j(k-1) + B_{j2} \cdot y_j(k-2) = C_{j1} \cdot x(k) - C_{j2} \cdot x(k-1) \quad , k=0,1,2...$$

for as many second order terms as there are (j=1,2,...), while the first order terms are realized by the equations:

$$y_j(k) - B_j \cdot y_j(k-1) = C_j \cdot x(k) \quad , k=0,1,2... \quad , B_j=A_j, \; C_j=K$$

again for as many j as there are first order terms.

The outputs $y_j(k)$ are then summed to give the filter output $y(k)$ (for each k). Note that this is justified by the linearity of the z-transformation.

The filter constructed above is called a recursive numerical filter (RNF), where the term "recursive" refers to the fact that past outputs contribute algebraically to the present output.

PROGRAM INFORMATION

The computations above have been implemented in a series of subroutines that are called by the driver subroutine BTRRNF. BTRRNF computes the coefficients of a specified filter and passes them to a routine which actually performs filtering operations.

BTRRNF is called by the FORTRAN statement:

```
CALL BTRRNF(NPOLES,ITYPE,CF,BW,T,IFERR)
```

VARIABLE	PURPOSE	TYPE	INPUT/OUTPUT
NPOLES	The number of poles in LP version, where NPOLES≤10	INTEGER	Input
ITYPE	A switch which determines whether a LP or BP filter will be generated. If ITYPE=0, then LP, If ITYPE=1, then BP. Note that shifting from LP to BP doubles the actual number of poles (see below)	INTEGER	Input
CF	The center frequency in Hz	REAL	Input
BW	The bandwidth in Hz	REAL	Input
T	The sampling period in secs	REAL	Input

IFERR	An error indicator If NPOLES .GT. 10, an error exists, IFERR returns as 0. If all goes well, IFERR returns as 1	INTEGER	Output

TABLE 1

The filter is specified in the call, and if IFERR is returned as 1, the filter is loaded in the filtering subroutine.

If the filter type is BP (ITYPE=1), a low pass filter with the appropriate bandwidth is set up and then bandshifted to produce the BP filter by substituting $(CF)\ /s'+s'/(cf)$ for s in the complex frequency plane. This is the operation which doubles the number of poles as mentioned in table 1. Note also that the center frequency is the geometric mean of the 3 dB points, not the algebraic mean.

As mentioned above, a call to BTRRNF computes the coefficients of the appropriate filter and passes them to a subroutine called RNF. This is the filtering routine and may be called by using the FORTRAN statement:

CALL RNF(X,Y,N)

VARIABLE	PURPOSE	TYPE	INPUT/OUTPUT
X	The input array containing the data to be filtered	REAL array dimension X(N)	Input
Y	The output array to contain the filtered data	REAL array dimension Y(N)	Output

N	Number of points to be filtered and size of X and Y arrays	INTEGER	Input

Table II

For user's purposes, this subroutine can be treated as a black box which returns one output value for each input value. By its nature, it "remembers" previous inputs, even after it has returned control to the main program. The user need not filter all the data in a time series at one time, if, for example, the data are arranged in several records on a tape. Consecutive calls to RNF using part of the series at a time (in the right order, of course) will give the same results as if all the filtering were done in one call. The following calls give identical results:

```
CALL RNF(X,Y,1000)
```

and

```
CALL RNF(X,Y,500)
CALL RNF(X(501), Y(501), 500)
```

The first is slightly more economical, but the computed results are the same.

It follows that if the filter remembers previous inputs, there should be a way to make it forget them, in order to pass unrelated series through the same filter. This capability is provided in a call which brings the filter to a fully relaxed state. This call, in FORTRAN, is:

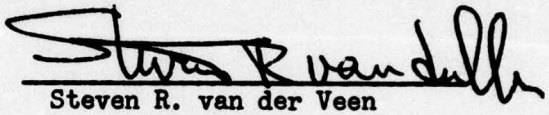
```
CALL ZERRNF
```

which zeros the memory elements of the filter.

BTRRNF and associated subroutines are located on FASTRAND file S1787/LIBRY and are available for general use.

SUMMARY

The subroutine BTRRNF allows the simulation of Butterworth filters on a digital computer. The method used is impulse-invariant in the sense that the computer output is identical to the sampled output of the equivalent analog filter. It is expected that the errors introduced by the use of a finite number system in the computer are much smaller than are the errors introduced in analog devices by variabilities in component specification.



Steven R. van der Veen
Physicist

REFERENCES

a. Freeman, H., Discrete Time Systems, an Introduction to the Theory, J. Wiley and Sons, N.Y., 1965

b. Rader, C. and Gold, B., "Digital Filter Design Techniques in the Frequency Domain", Proc. of the IEEE, Vol. 55, No. 2, Feb 1967

APPENDIX A: SAMPLE PROGRAM

A record on FASTRAND file SRV/82875 was known to contain a bottom reflected signal embedded in noise. Since it was known that the noise level was high outside the frequency band 100-400 Hz, it was decided to pass the signal through a bandpass filter with that passband. A program listing of the sample program and the plotted CALCOMP output (Figure 1) follow, demonstrating the use of the BTRRNF subroutine.

```
1*    C
2*    C       PROGRAM TO DEMONSTRATE DIGITAL FILTERING ROUTINE BTRRNF
3*    C
4*    C       THE FIRST RECORD ON FASTRAND FILE SRV/82875 WAS
5*    C       KNOWN TO CONTAIN A TIME SERIES WHICH WAS PREDOMINANTLY
6*    C       NOISE OUTSIDE THE FREQUENCY BAND 100-400 HZ.
7*    C       THEREFORE IT WAS USED HERE TO DEMONSTRATE THE USE OF
8*    C       THE BANDPASS FILTER.
9*    C
10*           DIMENSION X(2048),Y(2002),IBUF(4096)
11*    C
12*    C       INITIALIZE PLOTTING ROUTINE AND READ IN RECORD
13*    C
14*           CALL PLOTS(IBUF,4096,3)
15*           CALL PLOT(0.,0.,200)
16*           CALL PLOT(0.,1.5,-3)
17*           CALL FACTOR(.8)
18*           Y(1)=.005
19*           DO 3 I=2,2000
20*           Y(I)=Y(I-1)+.005
21*         3 CONTINUE
22*           Y(2001)=0.
23*           Y(2002)=1.
24*           CALL FTRAN(1,'82875')
25*           CALL FTRAN(3,'SRV',L)
26*           IF(L)99,99,
27*           ISECTR=1
28*           CALL FTRAN(8,'SRV',ISECTR,2048,X,IFERR)
29*         1 IF(IFERR+1)98,1,
30*    C
31*    C       SET UP A 5-POLE BP FILTER CENTERED AT 200 HZ, 300 HZ
32*    C       WIDE FOR A SAMPLING RATE OF 5000 HZ.
33*    C
34*           CALL BTRRNF(5,1,200.,300.,.0002,IFERR)
35*           IF(IFERR .NE. 1) GO TO 97
36*    C
37*    C       PLOT THE UNFILTERED SIGNAL
38*    C
39*           CALL AXIS(0.,0.,'SIGNAL AMPLITUDE',16,3.,90.,-.05,.05,10.)
40*           X(2001)=-.05
41*           X(2002)=.05
42*           CALL LINE(Y,X,2000,1,0,0)
43*           CALL SYMBOL(3.,2.5,.15,'UNFILTERED DATA',0.,15)
44*           CALL PLOT(0.,4.,-3)
45*    C
46*    C       FILTER 2000 POINTS OF THE X ARRAY AND PLOT THE OUTPUT
47*    C
48*           CALL RNF(X,X,2000)
49*           CALL AXIS(0.,0.,'SIGNAL AMPLITUDE',16,3.,90.,-.05,.05,10.)
50*           CALL LINE(Y,X,2000,1,0,0)
51*           CALL SYMBOL(3.,2.5,.15,'FILTERED DATA',0.,13)
52*           CALL PLOT(12.,-4.,-3)
53*    C
54*           STOP
55*        99 STOP 99
56*        98 STOP 98
57*        97 STOP 97
58*           END
```

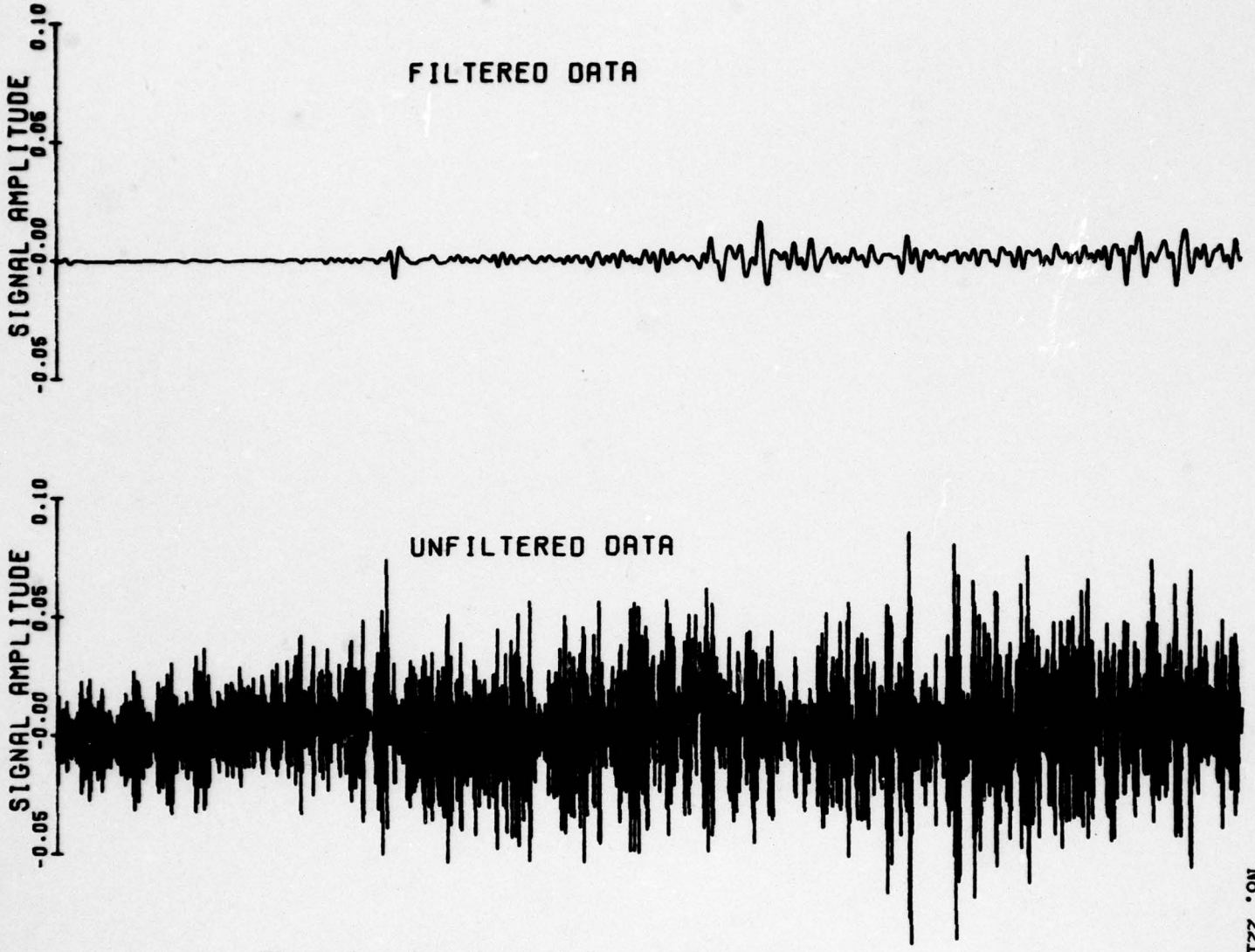


Figure 1: plot of output from sample program
time vs. signal amplitude

APPENDIX B: PROGRAM LISTINGS

All routines written for this program were written in FORTRAN V for the UNIVAC 1108 with the exception of the subroutine RNF, which was written in 1108 Assembly Language. Listings of all programs follow.

```
00101     1*          SUBROUTINE BTRRNF(NPOLES,ITYPE,CF,BW,T,IFERR)
00101     2*    C
00101     3*    C   ** SUBROUTINE BTRRNF EVALUATES THE COEFFICIENTS OF
00101     4*    C   ** A BUTTERWORTH FILTER.  THE PARAMETER ITYPE
00101     5*    C   ** CONTROLS THE TYPE OF FILTER (LP OR BP).  IF ITYPE
00101     6*    C   ** IS 0, A LOW PASS FILTER IS GENERATED.  IF IT
00101     7*    C   ** IS 1, A BAND PASS FILTER IS GENERATED.  NPOLES IS
00101     8*    C   ** THE NUMBER OF POLES IN THE LP VERSION.  THIS
00101     9*    C   ** VERSION IS BANDSHIFTED IF A BP FILTER IS DESIRED.
00101    10*    C   ** BW IS THE BANDWIDTH OF THE FILTER, AND CF IS THE
00101    11*    C   ** CENTER FREQUENCY (IF ONE IS NECESSARY).  T IS THE
00101    12*    C   ** SAMPLING INTERVAL.  ALL ARE IN UNITS OF HZ OR
00101    13*    C   ** SECS.  IFERR IS AN ERROR INDICATOR AND WILL BE
00101    14*    C   ** 1 ON A NORMAL RETURN.  NPOLES MUST BE .LE. 10.
00101    15*    C   ** IN ALL CASES, IF IFERR IS SET TO 1, A FILTER
00101    16*    C
00101    17*    C   ** HAS BEEN LOADED IN THE RNF FILTER SUBROUTINE.
00101    18*    C   ** SEE NUSC/NL TECH MEMO 2211-309-70.  'A PROGRAM
00101    19*    C   ** TO SIMULATE BUTTERWORTH FILTERS ON A DIGITAL
00101    20*    C   ** COMPUTER'
00101    21*    C
00103    22*         DOUBLE PRECISION POLE(2,5),POLES(2,10),ZERO(2,1),
00103    23*        *ZEROS(2,10),SCALE,TC


00103    24*    C
00103    25*    C   ** CONVERT NPOLES,T TO STORE IN PROGRAM
00103    26*    C
00104    27*         TC=T
00105    28*         N=NPOLES
00106    29*         N1=0
                                                          NUSC/NL Tech Memo
00106    30*    C                                         No. 2211-309-70
00106    31*    C   ** N MUST BE .LE. 10, TEST IT
00106    32*    C
00107    33*         IF(N .GT. 10) GO TO 100
00107    34*    C
00107    35*    C   ** LOCATE POLES OF N-POLE LP FILTER WITH BANDWIDTH BW
00107    36*    C
00111    37*         CALL BTRWRT(N,BW,POLE,SCALE,M1,M2)
00111    38*    C
00111    39*    C   ** IF ITYPE .EQ. 1, BANDPASS FILTER IS REQUESTED.
00111    40*    C   ** GENERATE IT AT STATEMENT 10
00111    41*    C
00112    42*         IF(ITYPE .NE. 0)  GO TO 10
00112    43*    C
00112    44*    C   ** ZERO ALL STORAGE IN RNF TO PREPARE IT TO
00112    45*    C   ** ACCEPT NEW FILTER COEFFS
00112    46*    C
00114    47*         CALL LODRNF
00114    48*    C
00114    49*    C   ** COMPUTE COEFFS OF FILTER AND PASS THEM TO RNF
00114    50*    C
00115    51*         CALL RECURS(ZERO,N1,POLE,M1,M2,SCALE,TC)
00115    52*    C
00115    53*    C   ** ALL O.K.  RETURN
00115    54*    C
00116    55*         IFERR=1
00117    56*         RETURN
```

```
00117     57*    C
00117     58*    C    ** CONVERT FROM LOWPASS TO BANDPASS IN S-DOMAIN
00117     59*    C
00120     60*       10 CALL LPTOBP(CF,POLE,M1,M2,ZERO,N1,POLES,M3,M4,
00120     61*         *ZEROS,N2,SCALE,IFRR)
00120     62*    C
00120     63*    C    ** TEST FOR ERROR IN LPTOBP
00120     64*    C    ** IF IFRR IS 0, AN ERROR EXISTS
00120     65*    C
00121     66*           IF(IFRR .EQ. 0) GO TO 100
00121     67*    C
00121     68*    C    ** ZERO ALL STORAGE IN RNF TO PREPARE IT TO
00121     69*    C    ** ACCEPT NEW FILTER COEFFS
00121     70*    C
00123     71*           CALL LODRNF
00123     72*    C
00123     73*    C    ** COMPUTE COEFFS OF FILTER AND PASS THEM TO RNF
00123     74*    C
00124     75*           CALL RECURS(ZEROS,N2,POLES,M3,M4,SCALE,TC)
00124     76*    C
00124     77*    C    ** ALL O.K.  RETURN
00124     78*    C
00125     79*           IFERR=1
00126     80*           RETURN
00126     81*    C


00126     82*    C    ** ERROR  RETURN WITH IFERR SET TO 0
00126     83*    C
00127     84*       100 IFERR=0
00130     85*           RETURN
00131     86*           END
```

```
      SUBROUTINE RECURS(ZERO,N,POLE,M1,M2,SCALE,T)
C
C     ** THIS SUBROUTINE PERFORMS THE IMPULSE-INVARIANT TRANSFORM FROM THE
C     ** S-PLANE TO THE Z-PLANE.  DOUBLE PRECISION IS REQUIRED FOR
C     ** ACCURACY.  NOTE THAT RECURS WILL NOT HANDLE MULTIPLE POLES.
C
C     ** THE ZEROS OF THE FILTER ARE TO BE N IN NUMBER, AND STORED IN ZERO(I,J)
C     ** WHERE ZERO(1,J) IS THE REAL PART OF THE J-TH ZERO, AND ZERO(2,J) IS THE
C     ** IMAGINARY PART OF THE J-TH ZERO.
C     ** THE POLES OF THE FILTER ARE TO BE STORED IN POLE(I,J) WHERE I HAS THE
C     ** SAME MEANING AS BEFORE.  IF J IS LESS THAN OF EQUAL TO M1, THE POLE IS A
C     ** REAL POLE AND ITS IMAGINARY PART WILL BE ZERO.  IF J IS GREATER THEN M1
C     ** THE POLE WILL BE ONE OF A COMPLEX PAIR.  ONLY ONE OF A COMPLEX PAIR IS
C     ** INCLUDED IN THE POLE ARRAY, THE OTHER BEING PRESENT BY IMPLICATION.
C     ** THERE ARE TO BE M1 REAL POLES AND M2 CONJUGATE POLE PAIRS.  SCALE IS
C     ** THE FILTER AMPLIFICATION CONSTANT, AND T IS THE SAMPLE INTERVAL.  ALL
C     ** NON-INTEGER VARIABLES ARE TO BE DOUBLE PRECISION EXCEPT FOR THE
C     ** ARRAY CONTAINING THE COMPUTED COEFFICIENTS FOR THE RNF ROUTINE.
C
C     ** SEE NUSC/NL TECH MEMO 2211-309-70,  'A COMPUTER PROGRAM
C     ** TO SIMULATE BUTTERWORTH FILTERS ON A DIGITAL COMPUTER'
C
C
      DOUBLE PRECISION T,SCALE,TBAR
      DOUBLE PRECISION Z(2),P(2),TEMP,A(2),POLE(2,20),ZERO(2,20),TMP(2)
      REAL COEFF(4)
      MTOT=M1+M2
      DO 100 II=1,MTOT
      Z(1)=1.0D0
      P(1)=1.0D0
      P(2)=0.0D0
      Z(2)=0.0D0
C
C     ** TEST FOR NO ZEROS
C
      IF(N .EQ. 0) GO TO 111
C
C     ** COMPUTE PRODUCT OF (POLE(II)-ZERO(I)) FOR ALL I AND STORE IN Z
C
      DO 110 I=1,N
      TEMP=Z(1)*(POLE(1,II)-ZERO(1,I))- Z(2)*(POLE(2,II)-ZERO(2,I))
      Z(2)=Z(2)*(POLE(1,II)-ZERO(1,I))+ Z(1)*(POLE(2,II)-ZERO(2,I))
  110 Z(1)=TEMP
```

```
C
C     ** COMPUTE PRODUCT OF (POLE(II)-POLE(I)) FOR ALL I .NE. II AND STORE
C     ** IN P
C
  11  DO 120 I=1,MTOT
      IF(I .EQ. II) GO TO 129
      TEMP=P(1)*(POLE(1,II)-POLE(1,I)) - P(2)*(POLE(2,II)-POLE(2,I))
      P(2)=P(2)*(POLE(1,II)-POLE(1,I)) + P(1)*(POLE(2,II)-POLE(2,I))
      P(1)=TEMP
C
C     ** TEST FOR REAL POLE
C
  129 IF(I .LE. M1) GO TO 130
C
C     ** CONJUGATE POLE PAIR.  COMPUTE TERM FOR CONJUGATE OF POLE(I)
C
      TEMP=P(1)*(POLE(1,II)-POLE(1,I)) - P(2)*(POLE(2,II)+POLE(2,I))
      P(2)=P(2)*(POLE(1,II)-POLE(1,I)) + P(1)*(POLE(2,II)+POLE(2,I))
      P(1)=TEMP
  130 CONTINUE
  120 CONTINUE
C
C     ** LOCATE POLE IN Z-PLANE
C
      A(1)= DEXP(POLE(1,II)*T)*DCOS(POLE(2,II)*T)
      A(2)= DEXP(POLE(1,II)*T)*DSIN(POLE(2,II)*T)
C
C     ** COMPUTE  Z/P*SCALE*T.  AMPLIFICATION CONSTANT
C
      TEMP=P(1)*P(1)+P(2)*P(2)
      TBAR=SCALE/TEMP*T
      TMP(1)= TBAR*(P(1)*Z(1)+P(2)*Z(2))
      TMP(2)= TBAR*(P(1)*Z(2)-Z(1)*P(2))
C
C     ** TEST FOR REAL POLE
C
      IF(II .LE. M1) GO TO 150
C


C     ** COMPLEX POLE--SECOND ORDER FILTER
C
      COEFF(1)=2.*TMP(1)
      COEFF(2)=2.*(A(1)* TMP(1)+A(2)* TMP(2))
      COEFF(3)=2.*A(1)
      COEFF(4)=A(1)*A(1)+A(2)*A(2)
C
C     ** LOAD COEFFICIENTS
C
      CALL ORD2(COEFF(1),COEFF(2),COEFF(3),COEFF(4))
      GO TO 100
C
C     ** REAL POLE--FIRST ORDER FILTER
C
  150 COEFF(1)=TMP(1)
      COEFF(2)=A(1)
C
C     ** LOAD COEFFICIENTS
C
      CALL ORD1(COEFF(1),COEFF(2))
  100 CONTINUE
      RETURN
      END
```

```
00101     1*          SUBROUTINE LPTOBP(CF,P1,M1,M2,Z1,N1,P2,M3,M4,
00101     2*         *Z2,N2,SC,IFERR)
00101     3*    C
00101     4*    C   ** THIS SUBROUTINE PERFORMS THE BANDSHIFT NECESSARY
00101     5*    C   ** TO CONVERT  A POLE-ZERO SET CENTERED ON THE REAL
00101     6*    C   ** AXIS TO A PAIR OF SETS CENTERED  AT +-(CF).  IT
00101     7*    C   ** DOES THIS BY THE TRANSFORM S=SPRIME/CF+CF/SPRIME.
00101     8*    C   ** THE POLES AND ZEROS TO BE TRANSFORMED ARE IN P1
00101     9*    C   ** AND Z1 AND THE RESULTS OF THE TRANSFORM ARE IN
00101    10*    C   ** P2 AND Z2.  NOTE THAT EACH POLE PRODUCES 2
00101    11*    C   ** POLES AND 1 ZERO(AT THE ORIGIN), WITH THE OPPOSITE
00101    12*    C   ** OCCURING FOR EACH ZERO.
00101    13*    C
00101    14*    C   ** SEE NUSC/NL TECH MEMO 2211-309-70.  'A PROGRAM
00101    15*    C   ** TO SIMULATE BUTTERWORTH FILTERS ON A DIGITAL
00101    16*    C   ** COMPUTER'
00101    17*    C
00103    18*          DOUBLE PRECISION P1(2,10),P2(2,10),Z1(2,5),Z2(2,5),
00103    19*         *A,B,C,D,R,PHI,PI,CFT
00104    20*          PI=3.14159265358979368D0
00105    21*          CFT=CF*PI*2.0D0
                                                    NUSC/NL Tech Memo
                                                    No. 2211-309-70

00106    22*          N2=0
00107    23*          M3=0
00110    24*          M4=0
00111    25*          IFERR=1
00111    26*    C
00111    27*    C   ** MM=NUMBER OF POLES
00111    28*    C   ** NN=NUMBER OF ZEROS
00111    29*    C
00112    30*          MM=M1+2*M2
00113    31*          NN=N1
00113    32*    C
00113    33*    C   ** EACH POLE PRODUCES A ZERO AT THE ORIGIN AND EACH
00113    34*    C   ** ZERO A POLE.  THESE WILL CANCEL.  COMPUTE THE
00113    35*    C   ** DIFFERENCE IN NUMBER.  IF MULTIPLE POLES OCCUR
00113    36*    C   ** AT THE ORIGIN (IF MM-NN .LT. -1), AN ERROR WILL
00113    37*    C   ** OCCUR IN RECURS, SO TEST MM-NN.
00113    38*    C
00114    39*          II=MM-NN
00115    40*          IF(II)1,7,5
00120    41*        1 IF(II+1)2,3,3
00120    42*    C
00120    43*    C   ** TOO MANY ZEROS
00120    44*    C
00123    45*        2 IFERR=0
00123    46*    C
00123    47*    C   ** PUT POLES AT ORIGIN, EVEN IF THERE ARE MORE THAN
00123    48*    C   ** ONE
00123    49*    C
00124    50*        3 DO 4 I=-1,II,-1
00127    51*          M3=M3+1
00130    52*          P2(1,M3)=-1.0-05
00131    53*        4 P2(2,M3)=0.000
00133    54*          GO TO 7
```

```
00133     55*    C
00133     56*    C   ** MORE POLES THAN ZEROS, PUT ZEROS AT ORIGIN
00133     57*    C
00134     58*        5 DO 6 I=1,II
00137     59*          N2=N2+1
00140     60*          Z2(1,N2)=0.0D0
00141     61*        6 Z2(2,N2)=0.0D0
00143     62*        7 CONTINUE
00144     63*          IF(MM)11,11,
00147     64*          MTOT=M1+M2
00150     65*          ICNT=M3
00150     66*    C
00150     67*    C   ** SOLVE TRANSFORMATION EQUATION(CF/P2+P2/CF-P1(1)=0)
00150     68*    C   ** FOR NEW POLES
00150     69*    C
00151     70*          DO 10 I=1,MTOT
00154     71*          A=P1(1,I)
00155     72*          B=P1(2,I)
00156     73*          C=2.0D0*A*B
00157     74*          D=A*A-B*B-4.0D0*CFT*CFT
00160     75*          PHI=DATAN2(C,D)/2.0D0
00161     76*          R=DSQRT(DSQRT(C*C+D*D))
00162     77*          C=R*DCOS(PHI)/2.0D0
00163     78*          D=R*DSIN(PHI)/2.000
00164     79*          ICNT=ICNT+1

00165     80*          P2(1,ICNT)=A/2.0D0+C
00166     81*          P2(2,ICNT)=B/2.0D0+D
00166     82*    C
00166     83*    C   ** TEST FOR P1(I) NON-REAL
00166     84*    C
00167     85*          IF(I .GT. M1) GO TO 9
00167     86*    C
00167     87*    C   ** P1(I) IS REAL, SECOND ROOT CONJUGATE OF FIRST
00167     88*    C
00171     89*          M4=M4+1
00172     90*          GO TO 10
00172     91*    C
00172     92*    C   ** P1(I) COMPLEX, FIND SECOND ROOT.  (ROOTS DUE TO
00172     93*    C   ** CONJUGATE OF P1(I) ARE EQUAL TO THE CONJUGATE OF
00172     94*    C   ** ROOTS DUE TO P1(I))
00172     95*    C
00173     96*        9 M4=M4+1
00174     97*          ICNT=ICNT+1
00175     98*          P2(1,ICNT)=A/2.0D0-C
00176     99*          P2(2,ICNT)=B/2.0D0-D
00177    100*          M4=M4+1
00200    101*       10 CONTINUE
00202    102*       11 CONTINUE
00203    103*          IF(NN)21,21,
```

```
00203   104*    C
00203   105*    C    ** SOLVE TRANSFORMATION EQUATION(CF/Z2+Z2/CF-Z1(I)=0)
00203   106*    C    ** FOR NEW ZEROS
00203   107*    C
00206   108*          DO 20 I=1,NN
00211   109*          N2=N2+1
00212   110*          A=Z1(1,I)
00213   111*          B=Z1(2,I)
00214   112*          C=2.000*A*B
00215   113*          D=A*A-B*B-4.000*CFT*CFT
00216   114*          PHI=DATAN2(C,D)/2.0D0
00217   115*          R=DSQRT(DSQRT(C*C+D*D))
00220   116*          C=R*DCOS(PHI)/2.000
00221   117*          D=R*DSIN(PHI)/2.000
00222   118*          Z2(1,N2)=A/2.0D0+C
00223   119*          Z2(2,N2)=B/2.0D0+D
00224   120*          N2=N2+1
00225   121*          Z2(1,N2)=A/2.0D0-C
00226   122*          Z2(2,N2)=B/2.0D0-D
00227   123*       20 CONTINUE
00231   124*       21 CONTINUE
00232   125*          RETURN
00233   126*          END
```

```
00101     1*          SUBROUTINE BTRWRT(N,BW,POLES,SCALE,M1,M2)
00102     2*          DOUBLE PRECISION POLES(2,N),BWC,PI,SCALE,XIND
00103     3*    C
00103     4*    C   ** BTRWRT LOCATES THE POLES OF AN N-POLE BUTTERWORTH
00103     5*    C   ** FILTER WITH BANDWIDTH BW.
00103     6*    C
00103     7*    C   ** SEE NUSC/NL TECH MEMO 2211-309-70, 'A PROGRAM
00103     8*    C   ** TO SIMULATE BUTTERWORTH FILTERS ON A DIGITAL
00103     9*    C   ** COMPUTER'
00103    10*    C
00104    11*          PI=3.14159265358979368D0
00105    12*          SCALE=1.0D0
00106    13*          BWC=2.0D0*PI*BW
00107    14*          ICT=0
00110    15*          M1=0
00111    16*          IF(N/2*2 .EQ. N) GO TO 10
00113    17*          SCALE=BWC
00114    18*          ICT=1
00115    19*          M1=1
00116    20*          POLES(1,1)=-BWC
00117    21*          POLES(2,1)=0.0D0
00120    22*       10 M2=N/2
00121    23*          DO 20 I=1,M2
00124    24*          ICT=ICT+1
00125    25*          XIND=(N-1+2*I)*PI/N/2.0D0
00126    26*          POLES(1,ICT)=BWC*DCOS(XIND)
00127    27*          POLES(2,ICT)=BWC*DSIN(XIND)

00130    28*       20 SCALE=SCALE*BWC*BWC
00132    29*          RETURN
00133    30*          END
```

```
00100     1*    C
00100     2*    C    ** IMMED PROVIDES IMMEDIATE LINKS BETWEEN RECURS
00100     3*    C    ** AND RNF, AND CAN BE BYPASSED WITH APPROPRIATE
00100     4*    C    ** CALLS IF DESIRED, BY PROVIDING ENTRY POINTS
00100     5*    C    ** 'ORD1(A,B)' AND 'ORD2(A,B,C,D)' AND THEN STORING
00100     6*    C    ** A, B, C, AND D IN ARRAYS
00100     7*    C
00100     8*    C    ** SEE NUSC/NL TECH MEMO 2211-309-70, 'A PROGRAM
00100     9*    C    ** TO SIMULATE BUTTERWORTH FILTERS ON A DIGITAL
00100    10*    C    ** COMPUTER'
00100    11*    C
00101    12*           SUBROUTINE IMMED
00105    13*           ENTRY ORD1(A,B)
00105    14*           CALL ORDER1(A,B)
00106    15*           RETURN
00107    16*           ENTRY ORD2(A,B,C,D)
00111    17*           CALL ORDER2(A,B,C,D)
00112    18*           RETURN
00113    19*           END
```

```
@1 ASM RNF,RNF
.
. RNF IS THE PROGRAM THAT ACTUALLY IMPLEMENTS THE FILTER GENERATED BY
. SUBROUTINE RECURS.  IT CONTAINS ENTRY POINTS FOR LOADING COEFFICIENTS,
. FOR RELAXING THE FILTER, FOR ZEROING ALL COEFFICIENT STORAGE, AND FOR
. ACTUALLY FILTERING AN ARRAY OF POINTS.
. SEE NUSC/NL TECH MEMO 2211-309-70,  'A PROGRAM TO SIMULATE
. BUTTERWORTH FILTERS ON A DIGITAL COMPUTER'
.
 REGNAM
$(0)
XPAST       RES     1          . STORE DATA ON LOC CNTR 0
YY1         RES     10         . PREVIOUS VALUE OF INPUT
YY2         RES     10         . PREV VAL OF OUTPUT, 2ND ORD FILTS
                               . NEXT TO LAST VAL OF OUTPUT, 2ND
                               .    ORDER FILTS
YY11        RES     10         . PREV VAL OF OUTPUT, 1ST ORD FILTS
N1ORDE      RES     1          . NUMBER OF FIRST ORDER FILTERS
N2ORDE      RES     1          . NUMBER OF 2NU ORDER FILTERS
CC1         RES     10         . C-SUB-1 2ND ORDER FILTS
CC2         RES     10         . C-SUB-2 2ND ORDER FILTS
BB1         RES     10         . B-SUB-1 2ND ORD FILTS
BB2         RES     10         . B-SUB-2 2ND ORD FILTS
CC11        RES     10         . C-SUB-1 1ST ORD FILTS
BB11        RES     10         . B-SUB-1 1ST ORD FILTS
SAVREG      RES     4          . TEMP STORAGE TO SAVE REGISTERS
$(1)
RNF*                           . CODE UNDER LOC CNTR 1
                               . ENTRY FOR CALL RNF(X,Y,N)
            SX      X1,SAVREG  . SAVE REG X1
            SX      X2,SAVREG+1 . SAVE REG X2
            SX      X3,SAVREG+2 . SAVE REG X3
            SX      X4,SAVREG+3 . SAVE REG X4
            LX      X3,0,X11   . ADDR(X(1)) INTO X3
            LXI,XU  X3,1       . 1 INTO X3 INCREMENT
            LX      X4,1,X11   . ADDR(Y(1)) INTO X4
            LXI,XU  X4,1       . 1 INTO X4 INCREMENT
            LX      X2,*2,X11  . N INTO X2 FOR LOOPING
            ANX,XU  X2,1       . INITIALIZE X2
OUTLOO                         . OUTER LOOP FOR N FILTER OPS
            SZ      A0         . 0 INTO A0(Y-ACCUMULATOR)
            LX      X1,N2ORDE  . N2ORDE  INTO X1
            JGD     X1,LOOP2   . TEST X1 NOT 0 AND INIT FOR LOOP
            J       FIRSTO     . N2ORDE 0, RUN 1ST ORD FILTS
LOOP2                          . LOOP FOR 2ND ORD FILTS
            LA      A2,CC1,X1  . CC1-SUB-X1 INTO A2
            FM      A2,0,X3    . A2 TIMES X-SUB-X3 INTO A2
            SA      A2,A4      . A2 INTO A4
            LA      A2,CC2,X1  . CC2-SUB-X1 INTO A2
            FM      A2,XPAST   . A2 TIMES XPAST INTO A2
            FAN     A4,A2      . A4 MINUS A2 INTO A4
            LA      A2,YY2,X1  . YY2-SUB-X1 INTO A2
            FM      A2,BB2,X1  . A2 TIMES BB2-SUB-X1 INTO A2
            FAN     A4,A2      . A4 MINUS A2 INTO A4
            LA      A2,YY1,X1  . YY1-SUB-X1 INTO A2
            SA      A2,YY2,X1  . A2 INTO YY2-SUB-X1
            FM      A2,BB1,X1  . A2 TIMES BB1-SUB-X1 INTO A2
            FA      A4,A2      . A4 PLUS  A2 INTO A4
```

```
                SA      A4,YY1,X1       . A4 INTO YY1-SUB-X1
                FA      A0,A4           . A0 PLUS A4 INTO A0(Y ACCUMULATOR)
                JGD     X1,LOOP2        . NEXT 2ND ORD FILT, IF ANY
FIRSTO                                  . START OF CODE FOR 1ST ORD FILTS
                LX      X1,N1ORDE       . N1ORDE INTO X1
                JGD     X1,LOOP1        . TEST X1 NOT 0 AND INIT FOR LOOP
                J       STOREY          . N1ORDE 0, STORE Y
LOOP1                                   . LOOP FOR 1ST ORD FILTS
                LA      A2,CC11,X1      . CC11-SUB-X1 INTO A2
                FM      A2,0,X3         . A2 TIMES X-SUB-X3 INTO A2
                SA      A2,A4           . A2 INTO A4
                LA      A2,BB11,X1      . BB11-SUB-X1 INTO A2
                FM      A2,YY11,X1      . A2 TIMES YY11-SUB-X1 INTO A2
                FA      A4,A2           . A4 PLUS  A2 INTO A4
                SA      A4,YY11,X1      . A4 INTO YY11-SUB-X1
                FA      A0,A4           . A0 PLUS A4 INTO A0(Y ACCUMULATOR)
                JGD     X1,LOOP1        . NEXT 1ST ORD FILT, IF ANY
STOREY          LA      A4,0,*X3        . X-SUB-X3 INTO A4 THEN INCREMENT
                                        .   X3 FOR NEXT X VALUE
                SA      A4,XPAST        . A4 INTO XPAST
                SA      A0,0,*X4        . A0 INTO Y-SUB-X4 THEN INCREMENT
                                        .   X4 FOR NEXT Y
                JGD     X2,OUTLOO       . NEXT POINT FOR FILTERING
                LX      X1,SAVREG       . RESTORE X1
                LX      X2,SAVREG+1     . RESTORE X2
                LX      X3,SAVREG+2     . RESTORE X3
                LX      X4,SAVREG+3     . RESTORE X4
                J       4,X11           . RETURN
LODRNF*                                 . ENTRY FOR CALL LODRNF
                                        . ZEROS ALL STORAGE
                LX,XU   X12,92          . 92 INTO X12
                J       $+2
ZERRNF*                                 . ENTRY FOR CALL ZERRNF
                                        . ZEROS ALL NON-COEFF. STORAGE
                LX,XU   X12,30          . 30 INTO X12
LOOPZ                                   . LOOP ON ZEROING STEP
                SZ      XPAST,X12       . 0 INTO XPAST+X12
                JGD     X12,LOOPZ       . ZERO NEXT LOC, IF ANY
                J       1,X11           . RETURN
ORDER2*                                 . ENTRY FOR CALL ORDER2(A,B,C,D)
                                        . PASSES 2ND ORD FILT COEFFS
                SX      X1,SAVREG       . SAVE REG X1
                LX      X1,N2ORDE       . N2ORDE INTO X1
                LA      A0,*0,X11       . A INTO A0
                SA      A0,CC1,X1       . A0 INTO CC1-SUB-X1
                LA      A0,*1,X11       . B INTO A0
                SA      A0,CC2,X1       . A0 INTO CC2-SUB-X1
                LA      A0,*2,X11       . C INTO A0
                SA      A0,BB1,X1       . A0 INTO BB1-SUB-X1
                LA      A0,*3,X11       . D INTO A0
                SA      A0,BB2,X1       . A0 INTO BB2-SUB-X1
                AX,XU   X1,1            . INCREMENT X1
                SX      X1,N2ORDE       . STORE NEW N2ORDE
                LX      X1,SAVREG       . RESTORE X1
                J       5,X11           . RETURN
ORDER1*                                 . ENTRY FOR CALL ORDER1(A,B)
                                        . PASSES 1ST ORD FILT COEFFS
                SX      X1,SAVREG       . SAVE REG X1
```

```
LX      X1,N1ORDE       . N1ORDE INTO X1
LA      A0,*0,X11       . A INTO A0
SA      A0,CC11,X1      . A0 INTO CC11-SUB-X1
LA      A0,*1,X11       . B INTO A0
SA      A0,BB11,X1      . A0 INTO BB11-SUB-X1
AX,XU   X1,1            . INCREMENT X1
SX      X1,N1ORDE       . STORE NEW N1ORDE
LX      X1,SAVREG       . RESTORE X1
J       3,X11           . RETURN
END
```